Constructing and Sampling Graphs with a Prescribed Joint Degree Distribution

Isabelle Stanton and Ali Pinar

Abstract One of the most influential recent results in network analysis is that many natural networks exhibit a power-law or log-normal degree distribution. This has inspired numerous generative models that match this property. However, more recent work has shown that while these generative models do have the right degree distribution, they are not good models for real life networks due to their differences on other important metrics like conductance. We believe this is, in part, because many of these real-world networks have very different *joint degree distributions*, i.e. the probability that a randomly selected edge will be between nodes of degree k and l. Assortativity is a sufficient statistic of the joint degree distribution, and it has been previously noted that social networks tend to be assortative, while biological and technological networks tend to be disassortative.

We suggest understanding the relationship between network structure and the joint degree distribution of graphs is an interesting avenue of further research. An important tool for such studies are algorithms that can generate random instances of graphs with the same joint degree distribution. This is the main topic of this paper and we study the problem from both a theoretical and practical perspective. We provide an algorithm for constructing simple graphs from a given joint degree distribution, and a Monte Carlo Markov Chain method for sampling them. We also show that the state space of simple graphs with a fixed degree distribution is connected via *end point switches*. We empirically evaluate the mixing time of this Markov Chain by using experiments based on the autocorrelation of each edge. These experiments

Isabelle Stanton
UC Berkeley, Berkeley, CA, e-mail: isabelle@eecs.berkeley.edu

Ali Pinar
Sandia National Laboratories, Livermore, CA, e-mail: apinar@sandia.gov
This work was funded by the applied mathematics program at the United States Department of Energy and performed at Sandia National Laboratories, a multiprogram laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the United States Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

show that our Markov Chain mixes quickly on real graphs, allowing for utilization of our techniques in practice.

1 Introduction

Graphs are widely recognized as the standard modeling language for many complex systems, including physical infrastructure (e.g., Internet, electric power, water, and gas networks), scientific processes (e.g., chemical kinetics, protein interactions, and regulatory networks in biology starting at the gene levels through ecological systems), and relational networks (e.g., citation networks, hyperlinks on the web, and social networks). The broader adoption of the graph models over the last decade, along with the growing importance of associated applications, calls for descriptive and generative models for real networks. What is common among these networks? How do they differ statistically? Can we quantify the differences among these networks? Answering these questions requires understanding the topological properties of these graphs, which have lead to numerous studies on many “real-world” networks from the Internet to social, biological and technological networks [18].

Perhaps the most prominent theme in these studies is the skewed degree distribution; real-world graphs have a few vertices with very high degree and many vertices with small degree. There is some dispute as to the exact distribution, some have called it power-law [5, 18], some log-normal [4, 51, 41, 8], and but all agree that it is ‘heavy-tailed’ [17, 54]. The ubiquity of this distribution has been a motivator for many different generative models and is often used as a metric for the quality of the model. Models like preferential attachment [5], the copying model [31], the Barabasi hierarchical model [53], forest-fire model, the Kronecker graph model [33], geometric preferential attachment [19] and many more [34, 59, 11] study the expected degree distribution and use the results to argue for the strength of their method. Many of these models also match other observed features, such as small diameter or densification [28]. However, recent studies comparing the generative models with real networks on metrics like conductance [35], core numbers [13] and clustering coefficients [30] show that the models do not match other important features of the networks.

The degree distribution alone does not define a graph. McKay’s estimate [39] shows that there may be exponentially many graphs with the same degree distribution. However, models based on degree distribution are commonly used to compute statistically significant structures in a graph. For example, the modularity metric for community detection in graphs [43, 42] assumes a null hypothesis for the structure of a graph based on its degree distribution, namely that probability of an edge between vertex v_i and v_j is proportional to $d_i d_j$, where d_i and d_j represent the degrees of vertices v_i and v_j. The modularity of a group of vertices is defined by how much their structure deviates from the null hypothesis, and a higher modularity signifies a better community. The key point here is that the null hypothesis is solely based on its degree distribution and therefore might be incorrect. Degree distribution based models are also used to predict graph properties [40, 2, 15, 14, 16], benchmark [32], and analyze the expected run time of algorithms [7].

These studies improve our understanding of the relationship between the degree distribution and the structure of a graph. The shortcomings of these studies give insight into what other features besides the degree distribution would give us a better grasp of a graph's structure. For example, the degree assortativity of a network measure whether nodes attach to other similar or dissimilar vertices. This is not specified by the degree distribution, yet studies have shown that social networks tend to be assortative, while biological and technological networks tend to be dissortative [47, 46]. An example of recent work using assortativity is [30]. In this study, a high assortativity is assumed for connections that generate high clustering coefficients, and this, in addition to preserving the degree distribution, results in very realistic instances of real-world graphs. Another study that has looked at the joint degree distribution is dK-graphs [38]. They propose modeling a graph by looking at the distribution of the structure of all sized k subsets of vertices, where $d = 1$ are vertex degrees, $d = 2$ are edge degrees (the joint degree distribution), $d = 3$ is the degree distribution of triangles and wedges, and so on. It is an interesting idea, as clearly the nK distribution contains all information about the graph, but it is far too detailed as a model. At what d value does the additional information become less useful?

One way to enhance the results based on degree distribution is to use a more restrictive feature such as the *joint degree distribution*. Intuitively, if degree distribution of a graph describes the probability that a vertex selected uniformly at random will be of degree k then its joint degree distribution describes the probability that a randomly selected *edge* will be between nodes of degree k and l. We will use a slightly different concept, the joint degree matrix, where the total number of nodes and edges is specified, and the numbers of edges between each set of degrees is counted. Note that while the joint degree distribution uniquely defines the degree distribution of a graph up to isolated nodes, graphs with the same degree distribution may have very different joint degree distributions. We are not proposing that the joint degree distribution be used as a stand alone descriptive model for generating networks. We believe that understanding the relationship between the joint degree distribution and the network structure is important, and that having the capability to generate random instances of graphs with the same joint degree distribution will help enable this goal. Experiments on real data are valuable, but also drawing conclusions only based on a limited data may be misleading, as the graphs may all be biased the same way. For a more rigorous study, we need a sampling algorithm that can generate random instances in a reasonable time, which is the motivation of this work.

The primary questions investigated by this paper are: Given a joint degree distribution and an integer n, does the joint degree distribution correspond to a real labeled graph? If so, can one construct a graph of size n with that joint degree distribution? Is it possible to construct or generate a *uniformly random* graph with that same joint degree distribution? We address these problems from both a theoretical and from an empirical perspective. In particular, being able to uniformly sample graphs allows one to empirically evaluate which other graph features, like diameter, or eigenvalues, are correlated with the joint degree distribution.

Contributions

We make several contributions to this problem, both theoretically and experimentally. First, we discuss the necessary and sufficient conditions for a given joint degree vector to be graphical. We prove that these conditions are sufficient by providing a new constructive algorithm. Next, we introduce a new configuration model for the joint degree matrix problem which is a natural extension of the configuration model for the degree sequence problem. Finally, using this configuration model, we develop Markov Chains for sampling both pseudographs and simple graphs with a fixed joint degree matrix. A pseudograph allows multiple edges between two nodes and self-loops. We prove the correctness of both chains and mixing time for the pseudograph chain by using previous work. The mixing time of the simple graph chain is experimentally evaluated using autocorrelation.

In practice, Monte Carlo Markov Chains are a very popular method for sampling from difficult distributions. However, it is often very difficult to theoretically evaluate the mixing time of the chain, and many practitioners simply stop the chain after 5,000, 10,000 or 20,000 iterations without much justification. Our experimental design with autocorrelation provides a set of statistics that can be used as a justification for choosing a stopping point. Further, we show one way that the autocorrelation technique can be adapted from real-valued samples to combinatorial samples.

2 Related Work

The related work can be roughly divided into two categories: constructing and sampling graphs with a fixed degree distribution using sequential importance sampling or Monte Carlo Markov Chain methods, and experimental work on heuristics for generating random graphs with a fixed joint degree distribution.

The methods for constructing graphs with a given degree distribution are primarily either reductions to perfect matchings or sequential sampling methods. There are two popular perfect matching methods. The first is the *configuration model* [10, 1]: k mini-vertices are created for each degree k vertex, and all the mini-vertices are connected. Any perfect matching in the configuration graph corresponds to a graph with the correct degree distribution by merging all of the identified mini-vertices. This allows multiple edges and self-loops, which are often undesirable. See Figure 1. The second approach, the *gadget configuration model*, prevents multi-edges and self-loops by creating a gadget for each vertex. If v_i has degree d_i, then it is replaced with a complete bipartite graph (U_i, V_i) with $|U_i| = n-1-d_i$ and $|V_i| = n-1$. Exactly one node in each V_i is connected to each other V_j, representing edge (i, j) [26]. Any perfect matching in this model corresponds exactly to a simple graph by using the edges in the matching that correspond with edges connecting any V_i to any V_j. We use a natural extension of the first configuration model to the joint degree distribution problem.

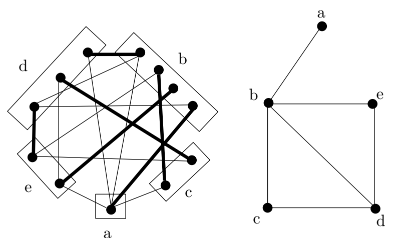


Fig. 1 On the left, we see an example of the configuration model of the degree distribution of the graph on the right. The edges corresponding to that graph are bold. Each vertex is split into a number of mini-vertices equal to its degree, and then all mini-vertices are connected. Not all edges are shown for clarity.

There are also sequential sampling methods that will construct a graph with a given degree distribution. Some of these are based on the necessary and sufficient Erdős-Gallai conditions for a degree sequence to be graphical [9], while others follow the method of Steger and Wormald [6, 57, 55, 24, 27]. These combine the construction and sampling parts of the problem and can be quite fast. The current best work can sample graphs where $d_{max} = O(m^{1/4-\tau})$ in $O(md_{max})$ time [6].

Another approach for sampling graphs with a given degree distribution is to use a Monte Carlo Markov Chain method. There is significant work on sampling perfect matchings [25, 12]. There has also been work specifically targeted at the degree distribution problem. Kannan, Tetali and Vempala [26] analyze the mixing time of a Markov Chain that mixes on the configuration model, and another for the gadget configuration model. Gkantsidis, Mihail and Zegura [21] use a Markov Chain on the configuration model, but reject any transition that creates a self-loop, multiple edge or disconnects the graph. Both of these chains use the work of Taylor [58] to argue that the state space is connected.

Amanatidis, Green and Mihail study the problems of when a given joint degree matrix has graphical representation and, further, when it has connected graphical representation [3]. They give necessary and sufficient conditions for both of these problems, and constructive algorithms. In Section 2, we give a simpler constructive algorithm for creating a graphical representation that is based on solving the degree sequence problem instead of alternating structures.

Another vein of related work is that of Mahadevan et al. who introduce the concept of dK-series [38, 37]. In this model, d refers to the dimension of the distribution and $2K$ is the joint degree distribution. They propose a heuristic for generating random $2K$-graphs for a fixed $2K$ distribution via edge rewirings. However, their method can get stuck if there exists a degree in the graph for which there is only 1 node with that degree. This is because the state space is not connected. We provide a theoretically sound method of doing this.

Finally, Newman also studies the problem of fixing an assortativity value, finding a *joint remaining degree distribution* with that value, and then sampling a random graph with that distribution using Markov Chains [47, 46]. His Markov Chain starts at any graph with the correct degree distribution and converges to a pseudograph with the correct joint remaining degree distribution. By contrast, our work provides a theoretically sound way of constructing a simple graph with a given joint degree distribution first, and our Markov Chain only has simple graphs with the same joint degree distribution as its state space.

3 Notation and Definitions

Formally, a degree distribution of a graph is the probability that a node chosen at random will be of degree k. Similarly, the joint degree distribution is the probability that a randomly selected *edge* will have end points of degree k and l. In this paper, we are concerned with constructing graphs that exactly match these distributions, so rather than probabilities, we will use a counting definition below and call it the *joint degree matrix*. In particular, we will be concerned with generating *simple* graphs that do not contain multiple edges or self-loops. Any graph that may have multiple edges or self loops will be referred to as a pseudograph.

Definition 1. The degree vector (DV) $d(G)$ of a graph G is a vector where $d(G)_k$ is the number of nodes of degree k in G.

A generic degree vector will be denoted by $\mathscr{D}$.

Definition 2. The joint degree matrix (JDM) $\mathscr{J}(G)$ of a graph G is a matrix where $\mathscr{J}(G)_{k,l}$ is exactly the number of edges between nodes of degree k and degree l in G.

A generic joint degree matrix will be denoted by $\mathscr{J}$. Given a joint degree matrix, $\mathscr{J}$, we can recover the number of edges in the graph as $m = \sum_{k=1}^{\infty}\sum_{l=k}^{\infty}\mathscr{J}_{k,l}$. We can also recover the degree vector as $\mathscr{D}_k = \frac{1}{k}(\mathscr{J}_{k,k} + \sum_{l=1}^{\infty}\mathscr{J}_{k,l})$. The term $\mathscr{J}_{k,k}$ is added twice because $k\mathscr{D}_k$ is the number of end points of degree k and the edges in $\mathscr{J}_{k,k}$ contribute two end points.

The number of nodes, n is then $\sum_{k=1}^{\infty}\mathscr{D}_k$. This count does not include any degree 0 vertices, as these have no edges in the joint degree matrix. Given n and m, we can easily get the degree distribution and joint degree distribution. They are $P(k) = \frac{1}{n}\mathscr{D}_k$ while $P(k,l) = \frac{1}{m}\mathscr{J}_{k,l}$. Note that $P(k)$ is not quite the marginal of $P(k,l)$ although it is closely related.

The Joint Degree Matrix Configuration Model

We propose a new configuration model for the joint degree distribution problem. Given $\mathscr{J}$ and its corresponding $\mathscr{D}$ we create k labeled mini-vertices for every vertex of degree k. In addition, for every edge with end points of degree k and l we create two labeled mini-end points, one of class k and one of class l. We connect all degree k mini-vertices to the class k mini-end points. This forms a complete bipartite graph for each degree, and each of these forms a connected component that is disconnected from all other components. We will call each of these components the "k-neighborhood". Notice that there are $k\mathscr{D}_k$ mini-vertices of degree k, and $k\mathscr{D}_k = \mathscr{J}_{k,k} + \sum_l \mathscr{J}_{k,l}$ corresponding mini-end points in each k-neighborhood. This is pictured in Figure 2. Take any perfect matching in this graph. If we merge each pair of mini-end points that correspond to the same edge, we will have some pseudograph that has exactly the desired joint degree matrix. This observation forms the basis of our sampling method.

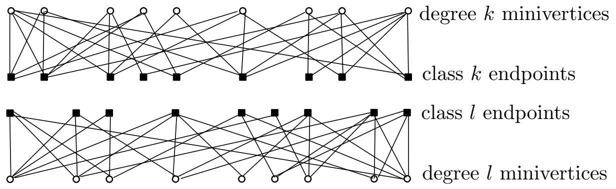


Fig. 2 The joint degree matrix configuration model. This shows just two degree neighborhoods of the joint degree matrix configuration model. Each vertex of degree k is split into k mini-vertices which are represented by the circles. These then form a complete bipartite component when they are connected with the class k end points, the squares. Each degree neighborhood is completely disconnected from all others. Not all edges are included for clarity.

4 Constructing Graphs with a Given Joint Degree Matrix

The Erdős-Gallai condition is a necessary and sufficient condition for a degree sequence to be realizable as a simple graph.

Theorem 1. Erdős-Gallai *A degree sequence $\overline{d} = \{d_1, d_2, \cdots d_n\}$ sorted in non-increasing order is graphical if and only if for every $k \le n$, $\sum_{i=1}^{k} d_i \le k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$.*

The necessity of this condition comes from noting that in a set of vertices of size k, there can be at most $\binom{k}{2}$ internal edges, and for each vertex v not in the subset, there can be at most $\min\{d(v), k\}$ edges entering. The condition considers each subset of decreasing degree vertices and looks at the degree requirements of those nodes. If the requirement is more than the available edges, the sequence cannot be graphical. The sufficiency is shown via the constructive Havel-Hakimi algorithm [23, 22].

The existence of the Erdős-Gallai condition inspires us to ask whether similar necessary and sufficient conditions exist for a joint degree matrix to be graphical. The following necessary and sufficient conditions were independently studied by Amanatidis et al. [3].

Theorem 2. *Let $\mathscr{J}$ be given and $\mathscr{D}$ be the associated degree distribution. $\mathscr{J}$ can be realized as a simple graph if and only if (1) $\mathscr{D}_k$ is integer-valued for all k and (2) $\forall k,l$, if $k \neq l$ then $\mathscr{J}_{k,l} \leq \mathscr{D}_k\mathscr{D}_l$. For each k, $\mathscr{J}_{k,k} \leq \binom{\mathscr{D}_k}{2}$.*

The necessity of these conditions is clear. The first condition requires that there are an integer number of nodes of each degree value. The next two are that the number of edges between nodes of degree k and l (or k and k) are not more than the total possible number of k to l edges in a simple graph defined by the marginal degree sequences. Amanatidis et al. show the sufficiency through a constructive algorithm. We will now introduce a new algorithm that runs in $O(m)$ time.

The algorithm proceeds by building a nearly regular graph for each class of edges, $\mathscr{J}_{k,l}$. Assume that $k \neq l$ for simplicity. Each of the $\mathscr{D}_k$ nodes of degree k receives $\lfloor \mathscr{J}_{k,l}/\mathscr{D}_k \rfloor$ edges, while $\mathscr{J}_{k,l} \mod \mathscr{D}_k$ each have an extra edge. Similarly, the l degree nodes have $\lfloor \mathscr{J}_{k,l}/\mathscr{D}_l \rfloor$ edges, with $\mathscr{J}_{k,l} \mod \mathscr{D}_l$ having 1 extra. We can then construct a simple bipartite graph with this degree sequence. This can be done in linear time in the number of edges using queues as is discussed after Lemma 1. If $k = l$, the only differences are that the graph is no longer bipartite and there are $2\mathscr{J}_{k,k}$ end points to be distributed among $\mathscr{D}_k$ nodes. To find a simple nearly regular graph, one can use the Havel-Hakimi [22, 23] algorithm in $O(\mathscr{J}_{k,k})$ time by using the degree sequence of the graph as input to the algorithm.

We must show that there is a way to combine all of these nearly-regular graphs together without violating any degree constraints. Let $d = \langle d_1, d_2, \cdots d_n \rangle$ be the sorted non-increasing order degree sequence from $\mathscr{D}$. Let $\hat{d_v}$ denote the residual degree sequence where the residual degree of a vertex v is d_v minus the number of edges that currently neighbor v. Also, let $\hat{\mathscr{D}}_k$ denote the number of nodes of degree k that have non-zero residual degree, i.e. $\hat{\mathscr{D}}_k = \sum_{d_j=k} \mathbf{1}(\hat{d_j} \neq 0)$.

Algorithm 1 Greedy Graph Construction with a Fixed JDM, Input: $\mathscr{J}$, n, m, $\mathscr{D}$

```
 1: for $k = n \cdots 1$ and $l = k \cdots 1$ do
 2:     if $k \neq l$ then
 3:         Let $a = \mathscr{J}_{k,l} \mod \mathscr{D}_k$ and $b = \mathscr{J}_{k,l} \mod \mathscr{D}_l$
 4:         Let $x_1 \cdots x_a = \lfloor \frac{\mathscr{J}_{k,l}}{\mathscr{D}_k} \rfloor + 1$, $x_{a+1} \cdots x_{\mathscr{D}_k} = \lfloor \frac{\mathscr{J}_{k,l}}{\mathscr{D}_k} \rfloor$ and $y_1 \cdots y_b = \lfloor \frac{\mathscr{J}_{k,l}}{\mathscr{D}_l} \rfloor + 1$, $y_{b+1} \cdots y_{\mathscr{D}_l} =$
            $\lfloor \frac{\mathscr{J}_{k,l}}{\mathscr{D}_l} \rfloor$
 5:         Construct a simple bipartite graph $B$ with degree sequence $x_1 \cdots x_{\mathscr{D}_k}, y_1 \cdots y_{\mathscr{D}_l}$
 6:     else
 7:         Let $c = 2\mathscr{J}_{k,k} \mod \mathscr{D}_k$
 8:         Let $x_1 \cdots x_c = \lfloor \frac{2\mathscr{J}_{k,k}}{\mathscr{D}_k} \rfloor + 1$ and $x_{c+1} \cdots x_{\mathscr{D}_k} = \lfloor \frac{2\mathscr{J}_{k,k}}{\mathscr{D}_k} \rfloor$
 9:         Construct a simple graph $B$ with the degree sequence $x_1 \cdots x_{\mathscr{D}_k}$
10:     end if
11:     Place $B$ into $G$ by matching the nodes of degree $k$ with higher residual degree with $x_1 \cdots x_a$
        and those of degree $l$ with higher residual degree with $y_1 \cdots y_b$. The other vertices in $B$ can
        be matched in any way with those in $G$ of degree $k$ and $l$
12:     Update the residual degrees of each $k$ and $l$ degree node.
13: end for
```

To combine the nearly uniform subgraphs, we start with the largest degree nodes, and the corresponding largest degree classes. It is not necessary to start with the largest, but it simplifies the proof. First, we note that after every iteration, the joint degree sequence is still feasible if $\forall k, l, k \neq l$ $\hat{\mathcal{J}}_{k,l} \leq \hat{\mathcal{D}}_k \hat{\mathcal{D}}_l$ and $\forall k$ $\hat{\mathcal{J}}_{k,k} \leq \binom{\hat{\mathcal{D}}_k}{2}$.

We will prove that Algorithm 4 can always satisfy the feasibility conditions. First, we note a fact.

Observation 1 *For all k,* $\sum_l \hat{\mathcal{J}}_{k,l} + \hat{\mathcal{J}}_{k,k} = \sum_{d_j=k} \hat{d}_j$

This follows directly from the fact that the left hand side is summing over all of the k end points needed by $\hat{\mathcal{J}}$ while the right hand side is summing up the available residual end points from the degree distribution. Next, we note that if all residual degrees for degree k nodes are either 0 or 1, then:

Observation 2 *If, for all j such that $d_j = k$, $\hat{d}_j = 0$ or 1 then* $\sum_{d_j=k} \hat{d}_j = \sum_{d_j=k} \mathbf{1}(\hat{d}_j \neq 0) = \hat{\mathcal{D}}_k$.

Lemma 1. *After every iteration, for every pair of vertices u, v of any degree k,* $|\hat{d}_u - \hat{d}_v| \leq 1$.

Amanatidis et al. refer to Lemma 1 as the *balanced degree invariant*. This is most easily proven by considering the vertices of degree k as a queue. If there are x edges to be assigned, we can consider the process of deciding how many edges to assign each vertex as being one of popping vertices from the top of the queue and reinserting them at the end x times. Each vertex is assigned edges equal to the number of times it was popped. The next time we assign edges with end points of degree k, we start with the queue at the same position as where we ended previously. It is clear that no vertex can be popped twice without all other vertices being popped at least once.

Lemma 2. *The above algorithm can always greedily produce a graph that satisfies $\mathcal{J}$, provided $\mathcal{J}$ satisfies the initial necessary conditions.*

Proof. There is one key observation about this algorithm - it maximizes $\hat{\mathcal{D}}_k \hat{\mathcal{D}}_l$ by ensuring that the residual degrees of any two vertices of the same degree never differ by more than 1. By maximizing the number of available vertices, we can not get stuck adding a self-loop or multiple edge. From this, we gather that if, for some degree k, there exists a vertex j such that $\hat{d}_j = 0$, then for all vertices of degree k, their residuals must be either 0 or 1. This means that $\sum_{d_j=k} \hat{d}_j = \hat{\mathcal{D}}_k \geq \hat{\mathcal{J}}_{k,l}$ for every other l from Observation 2.

From the initial conditions, we have that for every k, l $\mathcal{J}_{k,l} \leq \mathcal{D}_k \mathcal{D}_l$. $\mathcal{D}_k = \hat{\mathcal{D}}_k$ provided that all degree k vertices have non-zero residuals. Otherwise, for any unprocessed pair, $\mathcal{J}_{k,l} \leq \min\{\hat{\mathcal{D}}_k, \hat{\mathcal{D}}_l\} \leq \hat{\mathcal{D}}_k \hat{\mathcal{D}}_l$. For the k, k case, it is clear that $\mathcal{J}_{k,k} \leq \hat{\mathcal{D}}_k \leq \binom{\hat{\mathcal{D}}_k}{2}$. Therefore, the residual joint degree matrix and degree sequence will always be feasible, and the algorithm can always continue. □

A natural question is that since the joint degree distribution contains all of the information in the degree distribution, do the joint degree distribution necessary conditions easily imply the Erdős-Gallai condition? This can easily be shown to be true.

Corollary 1. *The necessary conditions for a joint degree matrix to be graphical imply that the associated degree vector satisfies the Erdős-Gallai condition.*

5 Uniformly Sampling Graphs with Monte Carlo Markov Chain (MCMC) Methods

We now turn our attention to uniformly sampling graphs with a given graphical joint degree matrix using MCMC methods. We return to the joint degree matrix configuration model. We can obtain a starting configuration for any graphical joint degree matrix by using Algorithm 1. This configuration consists of one complete bipartite component for each degree with a perfect matching selected. The transitions we use select any end point uniformly at random, then select any other end point in its degree neighborhood and swap the two edges that these neighbor. In Figure 2, this is equivalent to selecting one of the square endpoints uniformly at random and then selecting another uniformly at random from the same connected component and then swapping the edges. A more complex version of this chain checks that this swap does not create a multiple edge or self-loop. Formally, the transition function is a randomized algorithm given by Algorithm 2.

Algorithm 2 Markov Chain Transition Function, Input: a configuration C

```
1: With probability 0.5, stay at configuration $C$. Else:
2: Select any endpoint $e_1$ uniformly at random. It neighbors a vertex $v_1$ in configuration $C$
3: Select any $e_2$ u.a.r from $e_1$'s degree neighborhood. It neighbors $v_2$
4: (Optional: If the graph obtained from the configuration with edges $E \cup \{(e_1,v_2),(e_2,v_1)\} \setminus \{(e_1,v_1),(e_2,v_2)\}$ contains a multi-edge or self-loop, reject)
5: $E \leftarrow E \cup \{(e_1,v_2),(e_2,v_1)\} \setminus \{(e_1,v_1),(e_2,v_2)\}$
```

There are two chains described by Algorithm 2. The first, $\mathscr{A}$ doesn't have step (4) and its state space is all pseudographs with the desired joint degree matrix. The second, $\mathscr{B}$ includes step (4) and only transitions to and from simple graphs with the correct joint degree matrix.

We remind the reader of the standard result that any irreducible, aperiodic Markov Chain with symmetric transitions converges to the uniform distribution over its state space. Both $\mathscr{A}$ and $\mathscr{B}$ are aperiodic, due to the self-loop to each state. From the description of the transition function, we can see that $\mathscr{A}$ is symmetric. This is less clear for the transition function of $\mathscr{B}$. Is it possible for two connected configurations to have a different number of feasible transitions in a given degree neighborhood? We show that it is not the case in the following lemma.

Lemma 3. *The transition function of $\mathscr{B}$ is symmetric.*

Proof. Let C_1 and C_2 be two neighboring configurations in $\mathscr{B}$. This means that they differ by exactly 4 edges in exactly 1 degree neighborhood. Let this degree be k and let these edges be e_1v_1 and e_2v_2 in C_1 whereas they are e_1v_2 and e_2v_1 in C_2. We want to show that C_1 and C_2 have exactly the same number of feasible k-degree swaps.

Without loss of generality, let e_x, e_y be a swap that is prevented by e_1 in C_1 but allowed in C_2. This must mean that e_x neighbors v_1 and e_y neighbors some $v_y \neq v_1, v_2$. Notice that the swap e_1e_x is currently feasible. However, in C_2, it is now infeasible to swap e_1, e_x, even though e_x and e_y are now possible.

If we consider the other cases, like e_x, e_y is prevented by both e_1 and e_2, then after swapping e_1 and e_2, e_x, e_y is still infeasible. If swapping e_1 and e_2 makes something feasible in C_1 infeasible in C_2, then we can use the above argument in reverse. This means that the number of feasible swaps in a k-neighborhood is invariant under k-degree swaps. □

The remaining important question is the connectivity of the state space over these chains. It is simple to show that the state space of $\mathscr{A}$ is connected. We note that it is a standard result that all perfect matchings in a complete bipartite graph are connected via edge swaps [58]. Moreover, the space of pseudographs can be seen exactly as the set of all perfect matchings over the disconnected complete bipartite degree neighborhoods in the joint degree matrix configuration model. The connectivity result is much less obvious for $\mathscr{B}$. We adapt a result of Taylor [58] that all graphs with a given degree sequence are connected via edge swaps in order to prove this. The proof is inductive and follows the structure of Taylor's proof.

Theorem 3. *Given two simple graphs, G_1 and G_2 of the same size with the same joint degree matrix, there exists a series of endpoint rewirings to transform G_1 into G_2 (and vice versa) where every intermediate graph is also simple.*

Proof. This proof will proceed by induction on the number of nodes in the graph. The base case is when there are 3 nodes. There are 3 realizable JDMs. Each is uniquely realizable, so there are no switchings available.

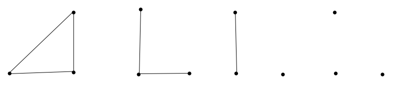


Fig. 3 The three potential joint degree distributions when $n = 3$.

Assume that this is true for $n = k$. Let G_1 and G_2 have $k+1$ vertices. Label the nodes of G_1 and G_2 $v_1 \cdots v_{k+1}$ such that $deg(v_1) \geq deg(v_2) \geq \cdots \geq deg(v_{k+1})$. Our goal will be to show that both graphs can be transformed in G_1' and G_2' respectively such that v_1 neighbors the same nodes in each graph, and the transitions are all through simple graphs. Now we can remove v_1 to create G_1'' and G_2'', each with $n-1$ nodes and identical JDMs. By the inductive hypothesis, these can be transformed into one other and the result follows.

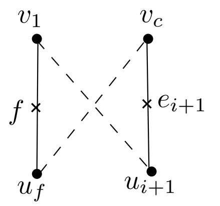


Fig. 4 The dotted edges represent the troublesome edges that we may need to swap out before we can swap v_1 and v_c.

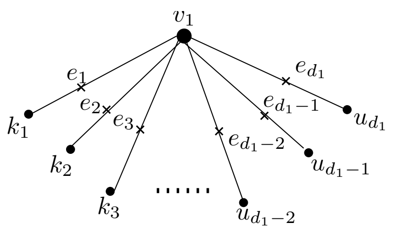


Fig. 5 The disk is v_1. The crosses are $e_1 \cdots e_{d_1}$.

We will break the analysis into two cases. For both cases, we will have a set of target edges, $e_1, e_2 \cdots e_{d_1}$ that we want v_1 to be connected to. Without loss of generality, we let this set be the edges that v_1 currently neighbors in G_2. We assume that the edges are ordered in reverse lexicographic order by the degrees of their endpoints. This will guarantee that the resulting construction for v_1 is graphical and that we have a non-increasing ordering on the requisite endpoints. Now, let k_i denote the endpoint in G_2 for edge e_i that isn't v_1.

Case 1) For the first case, we will assume that v_1 is already the endpoint of all edges $e_1, e_2 \cdots e_{d_1}$ but that all of the k_i may not be assigned correctly as in Figure 5. Assume that $e_1, e_2 \cdots e_{i-1}$ are all edges $(v_1, k_1) \cdots (v_1, k_{i-1})$ and that e_i is the first that isn't matched to its appropriate k_i.

Call the current endpoint of the other endpoint of e_i u_i. We know that $deg(k_i) = deg(u_i)$ and that k_i currently neighbors $deg(k_i)$ other nodes, $\Gamma(k_i)$. We have two cases here. One is that $v_1 \in \Gamma(k_i)$ but via edge f instead of e_i. Here, we can swap v_1 on the endpoints of f and e_i so that the edge $v_1 - e_i - k_i$ is in the graph. f can not be an e_j where $j < i$ because those edges have their correct endpoints, k_j assigned. This is demonstrated in Figure 6.

The other case is that $v_1 \notin \Gamma(k_i)$. If this is the case, then there must exist some $x \in \Gamma(k_i) \setminus \Gamma(u_i)$ because $d(u_i) = d(k_i)$ and u_i neighbors v_1 while k_i doesn't. Therefore, we can swap the edges $v_1 - e_i - u_i$ and $x - f - k_i$ to $v_1 - e_i - k_i$ and $x - f - u_i$ without creating any self-loops or multiple edges. This is demonstrated in Figure 6.

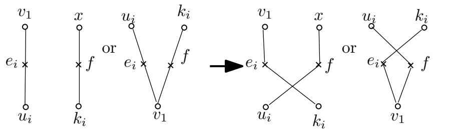


Fig. 6 The two parts of Case (1).

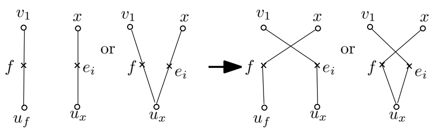


Fig. 7 The two parts of Case (2)

Therefore, we can swap all of the correct endpoints onto the correct edges.

Case 2) For the second case, we assume that the edges $e_1, \cdots e_{d_1}$ are distributed over l nodes of degree d_1. We want to show that we can move all of the edges $e_1 \cdots e_{d_1}$ so that v_1 is an endpoint. If this is achievable, we have exactly Case 1.

Let $e_1, \cdots e_{i-1}$ be currently matched to v_i and let e_i be matched to some x such that $deg(x) = d_1$. Let f be an edge currently matched to v_1 that is not part of $e_1 \cdots e_{d_1}$ and let its other endpoint be u_f. Let the other end point of e_i be u_x as in Figure 7.

We now have several initial cases that are all easy to handle. First, if v, x, u_x, u_f are all distinct and (v, u_x) and (x, u_f) are not edges then we can easily swap v and x such that the edges go from $v - f - u_f$ and $x - e_i - u_x$ to $v - e_i - u_x$ and $x - f - u_f$. Next, if $u_f = u_x$ then we can simply swap v_1 onto e_i and x onto f and, again, v_1 will neighbor e_i. This will not create any self-loops or multiple edges because the graph itself will be isomorphic. This situations are both shown in Figure 7.

The next case is that $x = u_f$. If we try to swap v_1 onto e_i then we create a self-loop from x to x via f. Instead, we note that since the JDM is graphical, there must exist a third vertex y of the same degree as v_1 and x that does not neighbor x. Now, y neighbors an edge g, and we can swap $x - f$ and $y - g$ to $x - g$ and $y - f$. The edges are $v_1 - f - y$ and $x - e_i - u_i$ and e_i can be swapped onto v_1 without conflict.

The cases left to analyze are those where the nodes are all distinct and (v_1, u_x) or (x, u_f) are edges in the graph. We will analyze these separately.

Case 2a) If (v_1, u_x) is an edge in the graph, then it must be so through some edge named g. Note that this means we have $v_1 - g - u_x$ and $x - e_i - u_x$. We can swap this to $v_1 - e_i - u_x$ and $x - g - u_x$ and have an isomorphic graph provided that g is not some e_j where $j < i$. This is the top case in Figure 8.

If g is some e_j then it must be that $u_x = k_j$. This is distinct from k_i. $deg(k_j) = deg(k_i)$ so there must exist some edge h that k_i neighbors with its other endpoint being y. There are again three cases, when $y \neq x, v_1$ $y = x$ and when $y = v_1$. These are the bottom three rows illustrated in Figure 8. The first is the simplest. Here, we can assume that k_j does not neighbor y (because it neighbors v_1 and x that k_i does not) so we can swap k_j onto h and k_i onto e_1. This has removed the offending edge, and we can now swap v_1 onto e_1 and x onto f.

When $y = x$, we first swap k_i onto e_j and k_j onto h. Next, we swap v onto e_i and x onto f as they no longer share an offending edge.

Finally, when $y = v_1$, we use a sequence of three swaps. The first is k_i onto e_j and k_j onto h. The next is v_1 onto e_1 and x onto h. Finally, we swap k_j back onto e_j and k_i onto e_i.

Case 2b) If (x, u_f) is an edge in the graph, then it must be through some edge g such that $x - g - u_f$ and $x - e_i - u_x$. Without loss of generality, assume that f is the only edge neighboring v_1 that isn't an e_j. Since f doesn't neighbor v_1 in G_2, there must either exist a w with $deg(w) = deg(u_f)$ or v_s with $deg(v_s) = d(v_1)$. This relies critically upon the fact that f and g are the same class edge. If there is a w, then it doesn't neighbor v_1 (or we can apply the above argument to find a w') and it must have some neighbor $y \in \Gamma(w) \setminus \Gamma(u)$ through edge h. Therefore, we can swap u_f onto h and w onto f. This removes the offending edge, and we can now swap v_1 onto e_i and x onto f.

If v_s exists instead, then by the same argument, there exists some edge h with endpoint u_s such that $v_s \notin \Gamma(u_f)$ and $u_s \notin \Gamma(x)$. Therefore, we can swap $v_s - h$ and $x - g$ to $v_s - g$ and $x - h$. This again removes the troublesome edge and allows us to swap v_1 onto e_i.

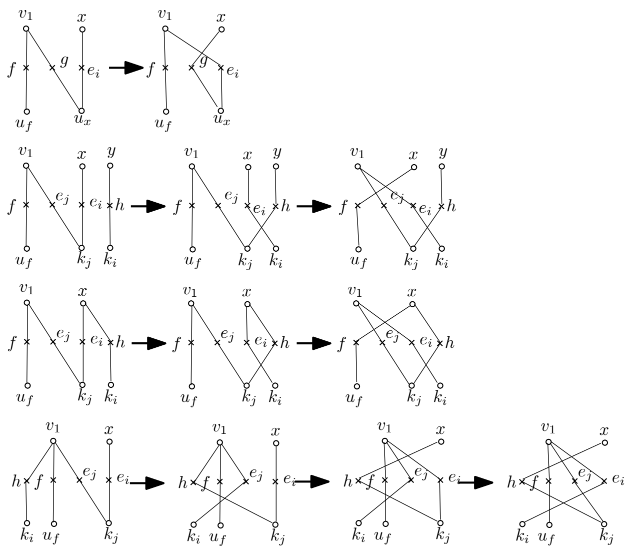


Fig. 8 A graphical representation of the situations discussed in Case (2a).

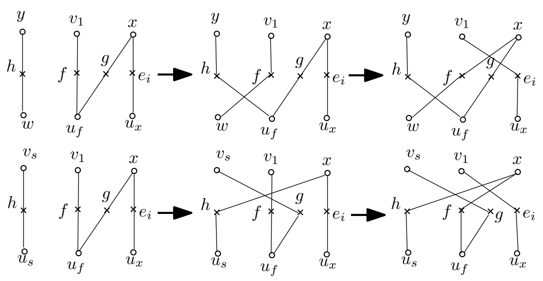


Fig. 9 A graphical representation of the situations discussed in Case (2b)

Therefore, given any node, a precise set of edges that it should neighbor, and a set of vertices that are the endpoints of those edges, we can use half-edge-rewirings to transform any graph G to G' that has this property, provided the set of edges is graphical. $\square$

Now that we have shown that both $\mathscr{A}$ and $\mathscr{B}$ converge to the uniform distribution over their respective state spaces, the next question is how quickly this happens. Note that from the proof that the state space of $\mathscr{B}$ is connected, we can upperbound the diameter of the state space by $3m$. The diameter provides a lower bound on the mixing time. In the next section, we will empirically estimate the mixing time to be also linear in m.

6 Estimating the Mixing Time of the Markov Chain

The Markov chain $\mathscr{A}$ is very similar to one analyzed by Kannan, Tetali and Vempala [26]. We can exactly use their canonical paths and analysis to show that the mixing time is polynomial. This result follows directly from Theorem 3 of [26] for chain $\mathscr{A}$. This is because the joint degree matrix configuration model can be viewed as $|\mathscr{D}|$ complete, bipartite, and disjoint components. These components should remain disjoint, so the Markov Chain can be viewed as a 'meta-chain' which samples a component and then runs one step of the Kannan, Tetali and Vempala chain on that component. Even though the mixing time for this chain is provably polynomial, this upper bound is too large to be useful in practice.

The analysis to bound the mixing time for $\mathscr{B}$ chain is significantly more complicated. One approach is to use the canonical path method to bound the congestion of this chain. The standard trick is to define a path from G_1 to G_2 that fixes the misplaced edges identified by $G_1 \oplus G_2$ in a globally ordered way. However, this is difficult to apply to chain $\mathscr{B}$ because fixing a specific edge may not be atomic, i.e. from the proof of Theorem 3 it may take up to 4 swaps to correctly connect a vertex with an endpoint if there are conflicts with the other degree neighborhoods. These swaps take place in other degree neighborhoods and are not local moves. Therefore, this introduces new errors that must be fixed, but can not be incorporated into $G_1 \oplus G_2$. In addition, step (4) also prevents us from using path coupling as a proof of the mixing time.

Given that bounding the mixing time of this chain seems to be difficult without new techniques or ideas, we use a series of experiments that substitute the *autocorrelation time* for the mixing time.

6.1 Autocorrelation Time

Autocorrelation time is a quantity that is related to the mixing time and is popular among physicists. We will give a brief introduction to this concept, and refer the reader to Sokal's lecture notes for further details and discussion [56].

The autocorrelation of a signal is the cross-correlation of the signal with itself given a lag t. More formally, given a series of data $\langle X_i \rangle$ where each X_i is a drawn from the same distribution X with mean μ and variance σ, the autocorrelation function is $R_X(t) = \frac{E[(X_i-\mu)(X_{i-t}-\mu)]}{\sigma^2}$.

Intuitively, the inherent problem with using a Markov Chain sampling method is that successive states generated by the chain may be highly correlated. If we were able to draw independent samples from the stationary distribution, then the autocorrelation of that set of samples with itself would go to 0 as the number of samples increased. The autocorrelation time is capturing the size of the gaps between sampled states of the chain needed before the autocorrelation of this 'thinned' chain is very small. If the thinned chain has 0 autocorrelation, then it must be exactly sampled from the stationary distribution. In practice, when estimating the autocorrelation from a finite number of samples, we do not expect it to go to exactly 0, but we do expect it to 'die away' as the number of samples and gap increases.

Definition 3. The exponential autocorrelation time is $\tau_{exp,X} = \limsup_{t\to\infty} \frac{t}{-\log|R_X(t)|}$ [56].

Definition 4. The integrated autocorrelation time is $\tau_{int,X} = \frac{1}{2}\sum_{t=-\infty}^{\infty} R_X(t) = \frac{1}{2} + \sum_{t=1}^{\infty} R_X(t)$ [56].

The difference between the exponential autocorrelation time and the integrated autocorrelation time is that the exponential autocorrelation time measures the time it takes for the chain to reach equilibrium after a cold start, or 'burn-in' time. The integrated autocorrelation time is related to the increase in the variance over the samples from the Markov Chain as opposed to samples that are truly independent. Often, these measurements are the same, although this is not necessarily true.

We can substitute the autocorrelation time for the mixing time because they are, in effect, measuring the same thing - the number of iterations that the Markov Chain needs to run for before the difference between the current distribution and the stationary distribution is small. We will use the integrated autocorrelation time estimate.

6.2 Experimental Design

We used the Markov Chain $\mathscr{B}$ in two different ways. First, for each of the smaller datasets, we ran the chain for 50,000 iterations 15 times. We used this to calculate the the autocorrelation values for each edge for each lag between 100 and 15,000 in multiples of 100. From this, we calculated the estimated integrated autocorrelation time, as well as the iteration time for the autocorrelation of each edge to drop under a threshold of 0.001. This is discussed in Section 6.4.

We also replicated the experimental design of Raftery and Lewis [52]. Given our estimates of the autocorrelation time for each size graph in Section 6.4, we ran the chain again for long enough to capture 10,000 samples where each sample had x iterations of the chain between them. x was chosen to vary from much smaller than the estimated autocorrelation time, to much larger. From these samples, we calculated the sample mean for each edge, and compared it with the actual mean from the joint degree matrix. We looked at the total variational distance between the sample means and actual means and showed that the difference appears to be converging to 0. We chose the mean as an evaluation metric because we were able to calculate the true means theoretically. We are unaware of another similarly simple metric.

We used the formulas for empirical evaluation of mixing time from page 14 of Sokal's survey [56]. In particular, we used the following:

- The sample mean is $\overline{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

- The sample unnormalized autocorrelation function is $\hat{C}(t) = \frac{1}{n-t}\sum_{i=1}^{n-t}(x_i - \overline{\mu})(x_{i+t} - \overline{\mu})$.
- The natural estimator of $R_X(t)$ is $\hat{\rho}(t) = \hat{C}(t)/\hat{C}(0)$
- The estimator for $\tau_{int,X}$ is $\hat{\tau}_{int} = \frac{1}{2}\sum_{t=-(n-1)}^{n-1}\lambda(t)\hat{\rho}(t)$ where λ is a 'suitable' cutoff function.

For a sequence of length x, calculating the autocorrelation of gap t requires $(x-t)^2$ dot products. Our experiments require that we calculate the autocorrelation for each possible edge in a graph for many lags. Thus running the full set of experiments requires $O(|V|^2 x \log x)$ time and is prohibitive when V is large. Note that x must necessarily be at least $\Theta(E)$ as well, since the mixing time can not be sub-linear in the number of edges. In Section 6.3 we will discuss results on the smaller datasets (AdjNoun, Dolphins, Football, Karate, and LesMis) that suggest a more feasible method for estimating autocorrelation time for larger graphs. We use this method to evaluate the autocorrelation time for the larger graphs as well, and present all of the results together. Rather than running the chain for 15,000 steps for the larger graphs, we selected more appropriate stopping conditions that were generally $10|E|$ based on the results for smaller graphs.

Data Sets

We have used several publicly available datasets, Word Adjacencies [48], Les Miserables [29], American College Football [20], the Karate Club [63], the Dolphin Social Network [36], C. Elegans Neural Network (celegans) [60, 62], Power grid (power) [61], Astrophysics collaborations (astro-ph) [44], High-Energy Theory collaborations (hep-th) [45], Coauthorships in network science (netscience) [49], and a snapshot of the Internet from 2006 (as-22july) [50]. In the following $|V|$ is the number of nodes, $|E|$ is the number of edges and $|\mathscr{J}|$ is the number of non-zero entries in the joint degree matrix.

Dataset	$\|E\|$	$\|V\|$	$\|\mathscr{J}\|$
AdjNoun	425	112	159
as-22july	48,436	22,962	5,496
astro-ph	121,251	16,705	11,360
celegans	2,359	296	642
Dolphins	159	62	61
Football	616	115	18
hep-th	15,751	8,360	629
Karate	78	34	40
LesMis	254	77	99
netscience	2,742	1,588	184
power	6,594	4,940	108

Table 1 Details about the datasets, $|V|$ is the number of nodes, $|E|$ is the number of edges and $|\mathscr{J}|$ is the number of unique entries in the $\mathscr{J}$.

6.3 Relationship Between Mean of an Edge and Autocorrelation

For each of the smaller graphs, AdjNoun, Dolphins, Football, Karate and LesMis, we ran the Markov Chain 10 times for 50,000 iterations and collected an indicator variable for each potential edge. For each of these edges, and each run, we calculated the autocorrelation function for values of t between 100 and 15,000 in multiples of 100. For each edge, and each run, we looked at the t value where the autocorrelation function first dropped below the threshold of 0.001. We then plotted the mean of these values against the mean of the edge, i.e. if it connects vertices of degree d_i and d_j (where $d_i \neq d_j$) then $\mu_e = \mathscr{J}_{d_i,d_j}/d_i d_j$ or $\mu_e = \mathscr{J}_{d_i,d_i}/\binom{d_i}{2}$ otherwise. The three most useful plots are given in Figures 10 and 11 as the other graphs did not contain a large range of mean values.

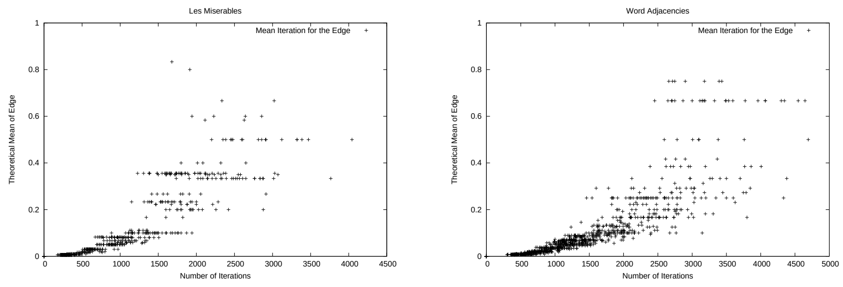
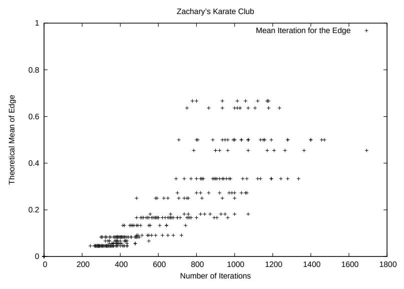


Fig. 10 The time for an edge's estimated autocorrelation function to pass under the threshold of 0.001 versus μ_e for that edge for LesMis and AdjNoun from L to R.

From these results, we identified a potential relationship between μ_e and the time to pass under a threshold. Unfortunately, none of our datasets contained a significant number of edges with larger μ_e values, i.e. between 0.5 and 1. In order to test this hypothesis, we designed a synthetic dataset that contained the many edges with values of μ_e at $\frac{i}{20}$ for $i = 1, \cdots 20$. We describe the creation of this dataset in the appendix.

The final dataset we created had 326 edges, 194 vertices and 21 distinct $\mathscr{J}$ entries. We ran the Markov Chain 200 times for this synthetic graph. For each run, we calculated the threshold value for each edge. Figure 11 shows the edges' mean vs its mean time for the autocorrelation value to pass under 0.001. We see that there is a roughly symmetric curve that obtains its maximum at $\mu_e = 0.5$.

This result suggests a way to estimate the autocorrelation time for larger graphs without repeating the entire experiment for every edge that could possibly appear. One can calculate μ_e for each edge from the JDM and sample edges with μ_e around 0.5. We use this method for selecting our subset of edges to analyze. In particular, we sampled about 300 edges from each of the larger graphs. For all of these except



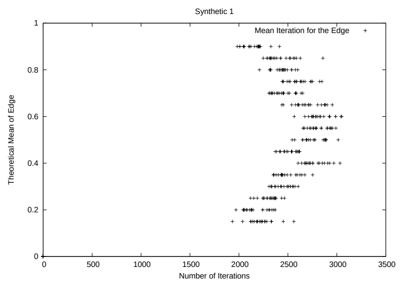


Fig. 11 The time for an edge's estimated autocorrelation function to pass under the threshold of 0.001 versus μ_e for that edge for Karate and the synthetic dataset. The synthetic dataset has a larger range of μ_e values than the real datasets and a significant number of edges for each value.

for power, the μ_e values were between 0.4 and 0.6. For power, the maximum μ_e value is about 0.15, so we selected edges with the largest μ values.

6.4 Autocorrelation Values

For each dataset and each run we calculated the unnormalized autocorrelation values. For the smaller graphs, this entailed setting t to every value between 100 and 15,000 in multiples of 100. We randomly selected 1 run for each dataset and graphed the autocorrelation values for each of the edges. We present the data for the Karate and Dolphins datasets in Figures 12 and 13. For the larger graphs, we changed the starting and ending points, based on the graph size. For example, for Netscience was analyzed from 2,000 to 15,000 in multiples of 100, while as-22july was analyzed from 1,000 to 500,000 in multiples of 1,000.

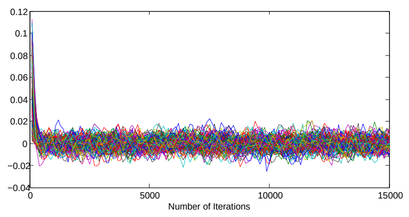


Fig. 12 The exponential drop-off for Karate appears to end after 400 iterations.

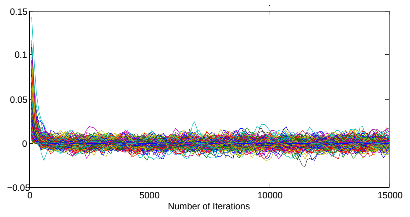


Fig. 13 The exponential drop-off for Dolphins appears to end after 600 iterations.

All of the graphs exhibit the same behavior. We see an exponential drop off initially, and then the autocorrelation values oscillate around 0. This behavior is due to the limited number of samples, and a bias due to using the sample mean for each edge. If we ignore the noisy tail, then we estimate that the autocorrelation 'dies off' at the point where the mean absolute value of the autocorrelation approximately converges, then we can locate the 'elbow' in the graphs. This estimate for all graphs is given in Table 3 at the end of this Section.

6.5 Estimated Integrated Autocorrelation Time

For each dataset and run, we calculated the estimated integrated autocorrelation time. For the datasets with fewer than 1,000 edges, we calculated the autocorrelation in lags of 100 from 100 to 15,000 for each dataset. For the larger ones, we used intervals that depended on the total size of the graph. We estimate $\hat{\rho}(t)$ as the size of the intervals times the sum of the values. The cut-off function we used for the smaller graphs was $\lambda(t) = 1$ if $0 < t < 15,000$ and 0 otherwise. This value was calculated for each edge. In Table 2 we present the mean, maximum and minimum estimated integrated autocorrelation time for each dataset over the runs of the Markov Chain using three different methods. For each of the edges, we first calculated the mean, median and max estimated integrated autocorrelation value over the various runs. Then, for each of these three values for each edge, we calculated the max, mean and min over all edges. For each of the graphs, the data series representing the median and max have each had their x-values perturbed slightly for clarity.

Dataset	$\|E\|$	mean	max	min	median	max	min	maximum	max	min
Karate	78	288.92	444.1	221.13	288.31	443	217.63	382.59	608.06	268.95
Dolphins	159	383.21	553.84	256.13	377.4	550.99	211.44	528.86	1134.1	397.35
LesMis	254	559.77	931.35	129.45	542.43	895.57	57.492	894.08	2598.6	342.76
AdjNoun	425	688.71	1154.9	156.49	659.06	1160.3	66.851	1186.1	4083.6	350.97
Football	616	962.42	2016.9	404.77	925.97	1646.9	349.12	1546.4	7514.3	967
celegans	2359	3340.2	4851.4	2458.8	3235.7	4861.4	2323.6	4844.6	7836.9	3065.5
netscience	2742	1791.4	3147.2	1087.7	1658.3	3033.2	937.8382	3401	7404	1894.7
power	6594	6624.5	17933	2166.9	4768.8	16901	250.6012	20599	54814	7074.7
hep-th	15751	26552	36816	14976	25608	37004	14130	46309	64936	25753
as-22july	48436	89637	139280	60627	87190	152490	58493	121930	256520	76214
astro-ph	121251	121860	298970	37706	119900	321730	46830	152930	408000	84498

Table 2 A summary of all the estimated integrated autocorrelation times. Mean refers to taking the mean autocorrelation time for each edge, and then the mean, min and max of these values over all measured edges. Similarly, median is the median value for each edge, while max is the maximum for each edge.

These values are graphed on a log-log scale plot. Further, we also present a graph showing the ratio of these values to the number of edges. The ratio plot, Figure 15, suggests that the autocorrelation time may be a linear function of the number of edges in the graph, however the estimates are noisy due to the limited number of runs.

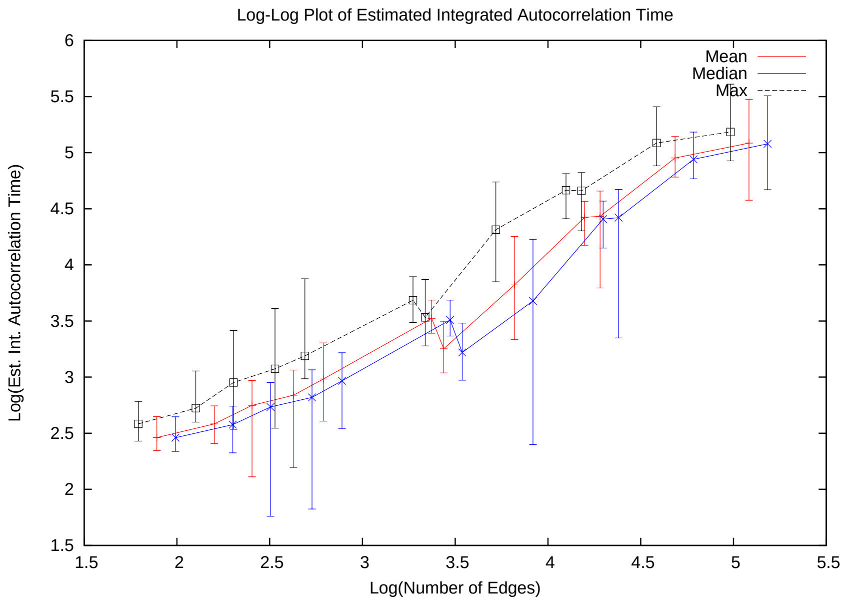


Fig. 14 The max, median and min values over the edges for the est. int. autocorrelation times in a log-log plot. L to R in order of size: Karate, Dolphins, LesMis, AdjNoun, Football, celegans, netscience, power, hep-th, as-22july and astro-ph

All three metrics give roughly the same picture. We note that there is much higher variance in estimated autocorrelation time for the larger graphs. If we consider the evidence of the log-log plot and the ratio plot, we suspect that the autocorrelation time of this Markov Chain is linear in the number of edges.

6.6 The Sample Mean Approaches the Real Mean for Each Edge

Given the results of the previous experiment estimating the integrated autocorrelation time, we next executed an experiment suggested by Raftery and Lewis [52]. First we note that for each edge e, we know the true value of $P(e \in G | G \text{ has } \mathscr{J})$ is exactly $\frac{\mathscr{J}_{k,l}}{\mathscr{D}_k \mathscr{D}_l}$ or $\frac{\mathscr{J}_{k,k}}{\binom{\mathscr{D}_k}{2}}$ if e is an edge between degrees k and l. This is because there are $\mathscr{D}_k \mathscr{D}_l$ potential (k,l) edges that show up in any graph with a fixed $\mathscr{J}$, and each graph has $\mathscr{J}_{k,l}$ of them. If we consider the graphs as being labeled, then we can see that each edge has an equal probability of showing up when we consider permutations of the orderings.

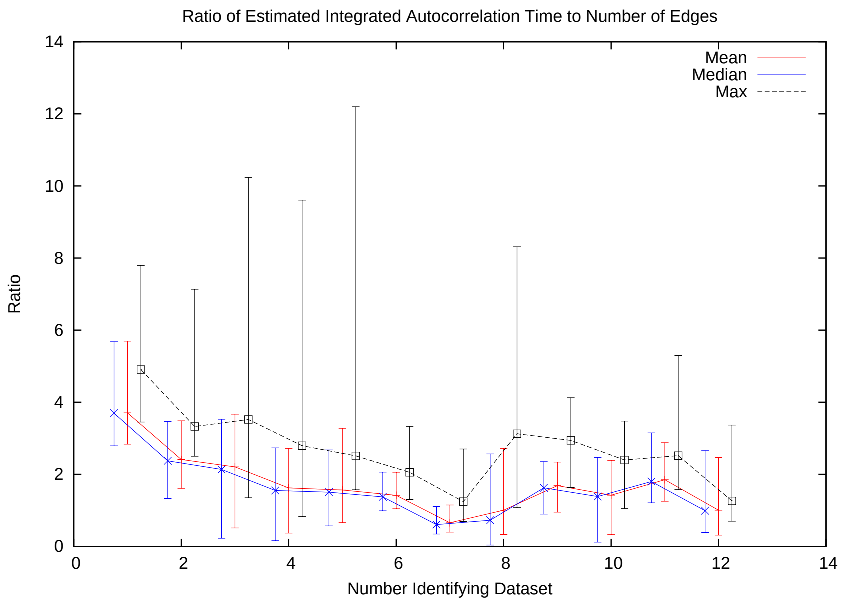


Fig. 15 The ratio of the max, median and min values over the edges to the number of edges for the estimated integrated autocorrelation times. L to R in order of size: Karate, Dolphins, LesMis, AdjNoun, Football, celegans, netscience, power, hep-th, as-22july and astro-ph

Thus, our experiment was to take samples at varying intervals, and consider how the sample mean of each edge compared with our known theoretical mean. For the smaller graphs, we took 10,000 samples at varying gaps depending on our estimated integrated autocorrelation time and repeated this 10 times. Additionally, we saw that the total variational distance quickly converged to a small, but non-zero value. We repeated this experiment with 20,000 samples and, for the two smallest graphs, Karate and Dolphins, we repeated the experiment with 5,000 and 40,000 samples. These results show that this error is due to the number of samples and not the sampler. For the graphs with more than 1,000 edges, each run resulted in 20,000 samples at varying gaps, and this was repeated 5 times. We present these results in Figures 18 through 28. If $S_{e,g}$ is the sample mean for edge e and gap g, and μ_e is the true mean, then the graphed value is $\sum_e |S_{e,g} - \mu_e| / \sum_e \mu_e$.

In all of the figures, the line runs through the median error for the runs and the error bars are the maximum and minimum values. We note that the maximum and minimum are very close to the median as they are within 0.05% for most intervals. These graphs imply that we are sampling uniformly after a gap of 175 for the Karate graph. For the dolphin graph, we see very similar results, and note that the error becomes constant after a sampling gap of 400 iterations.

For the larger graphs, we varied the gaps based on the graph size, and then focused on the area where the error appeared to be decreasing. Again, we see consistent results, although the residual error is higher. This is to be expected because

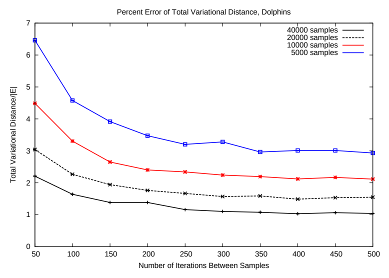


Fig. 16 The Dolphin Dataset with 5,000 to 40,000 samples

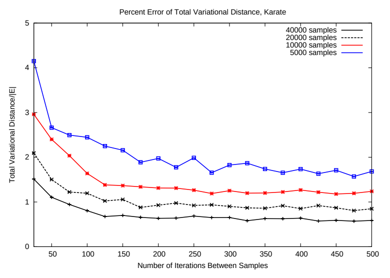


Fig. 17 The Karate Dataset with 5,000 to 40,000 samples

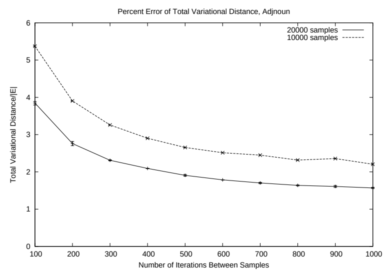


Fig. 18 The AdjNoun Dataset with 10,000 and 20,000 samples

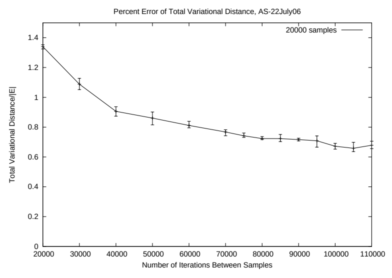


Fig. 19 The AS-22July06 Dataset with 20,000 samples

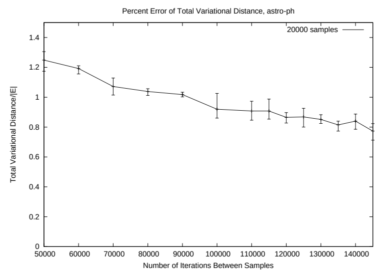


Fig. 20 The Astro-PH Dataset with 20,000 samples

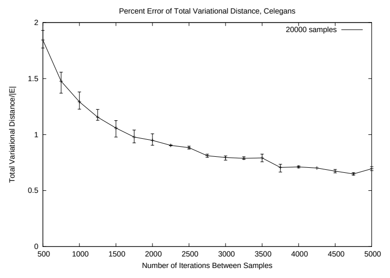


Fig. 21 The Celegans Dataset with 20,000 samples

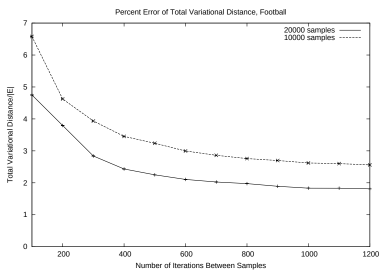


Fig. 22 The Football Dataset with 10,000 and 20,000 samples

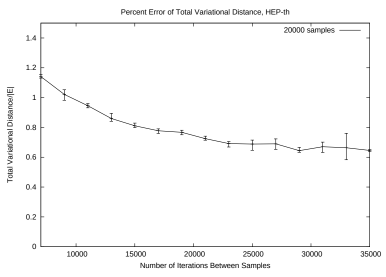


Fig. 23 The Hep-TH Dataset with 20,000 samples

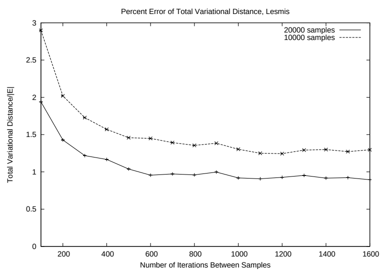


Fig. 24 The LesMis Dataset with 10,000 and 20,000 samples

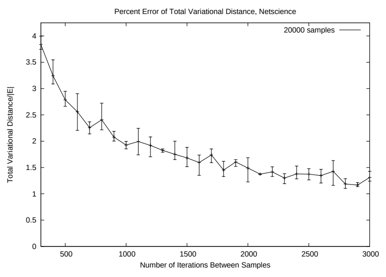


Fig. 25 The Netscience Dataset with 20,000 samples

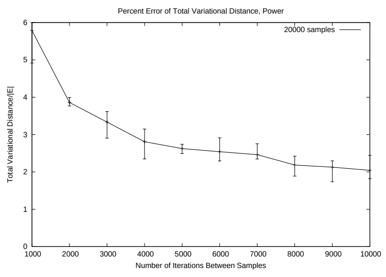


Fig. 26 The Power Dataset with 20,000 samples

there are more potential edges in these graphs, so we took relatively fewer samples per edge. A summary of the results can be found in Table 3.

6.7 *Summary of Experiments*

	$\|E\|$	Max EI	Mean Conv.	Thresh.
AdjNoun	425	1186	900	700
AS-22July	48,436	256,520	95,000	156,744
Astro-PH	121,251	408,000	120,000	343,154
Celegans	2,359	7836.9	3,750	7,691
Dolphins	159	528	400	600
Football	616	1546	1000	900
Hep-TH	15,751	64,936	28,000	22,397
Karate	78	382	175	400
LesMis	254	894	800	1000
Netscience	2,742	7,404	2,000	7,017
Power	6,594	54,814	8,000	7,270

Table 3 A summary of estimates on convergence from the three experiments. The values are the Maximum Estimated Integrated Autocorrelation time (Max EI, the third column of Table 2), the Sample Mean Convergence iteration number, and the time to drop under the Autocorrelation Threshold. The Autocorrelation threshold was calculated as when the average absolute value of the autocorrelation was less than 0.0001

Based on the results in this table, our recommendation would be that running the Markov Chain for $5m$ steps would satisfy all running time estimates except for Power's results for the Maximum Estimated Integrated Autocorrelation time. This estimate is significantly lower than the result for Chain $\mathscr{A}$ that was obtained using the standard theoretical technique of canonical paths.

7 Conclusions and Future Work

This paper makes two primary contributions. The first is the investigation of Markov Chain methods for uniformly sampling graphs with a fixed joint degree distribution. Previous work shows that the mixing time of $\mathscr{A}$ is polynomial, while our experiments suggest that the mixing time of $\mathscr{B}$ is also polynomial. The relationship between the mean of an edge and the autocorrelation values can be used to efficiently experiment with larger graphs by sampling edges with mean between 0.4 and 0.6 and repeating the analysis for just those edges. This was used to repeat the experiments for larger graphs and to provide further convincing evidence of polynomial mixing time.

Our second contribution is in the design of the experiments to evaluate the mixing time of the Markov Chain. In practice, it seems the stopping time for sampling is often chosen without justification. Autocorrelation is a simple metric to use, and can be strong evidence that a chain is close to the stationary distribution when used correctly.

Acknowledgments

The authors would like to acknowledge helpful contributions of David Gleich, Satish Rao, Jaideep Ray, Alistair Sinclair, Virginia Vassilevska Williams and Wes Weimer.

References

1. W. Aiello, F. R. K. Chung, and L. Lu. A random graph model for massive graphs. *STOC*, pages 171–180, 2000.
2. W. Aiello, F. R. K. Chung, and L. Lu. A random graph model for power law graphs. *Experimental Math*, 10:53–66, 2000.
3. Y. Amanatidis, B. Green, and M. Mihail. Graphic realizations of joint-degree matrices. *Manuscript*, 2008.
4. L. A. N. Amaral, A. Scala, M. Barthelemy, and H.E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97:11149–11152, 2000.
5. A-L Barabasi and R Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.
6. M. Bayati, J. H. Kim, and A. Saberi. A sequential algorithm for generating random graphs. *APPROX-RANDOM*, pages 326–340, 2007.
7. J. W. Berry, D. J. Nordman, C. A. Phillips, and A. Wilson. Listing triangles in expected linear time on a class of power law graphs. *Technical report, Sandia National Laboratories*, 2010.
8. Z. Bi, C. Faloutsos, and F. Korn. The dgx distribution for mining massive, skewed data. *KDD*, pages 17–26, 2001.
9. J. Blitzstein and P. Diaconis. A sequential importance sampling algorithm for generating random graphs with prescribed degrees, 2006.
10. B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European J. Comb*, 1:311–316, 1980.
11. B. Bollobás, O. Riordan, J. Spencer, and G. Tusndy. The degree sequence of a scale-free random graph process. *Random Structures & Algorithms*, 18 (3):279290, 2001.
12. A. Z. Broder. How hard is it to marry at random? (on the approximation of the permanent). *STOC*, pages 50–58, 1986.
13. A. Pinar C. Seshadri and T. G. Kolda. A hitchhiker's guide to choosing parameters of stochastic kronecker graphs. *http://arxiv.org/abs/1102.5046*, 2011.
14. F. R. K. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1), 2003.
15. Fan Chung, Linyuan Lu, , and Van Vu. Spectra of random graphs with given expected degrees. *PNAS*, 100:6313–6318, 2003.
16. F.R.K. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *ANNALS OF COMBINATORICS*, pages 125–145, 2002.
17. A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

18. M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*, pages 251–262, 1999.
19. A. Flaxman, A. Frieze, and J. Vera. A geometric preferential attachment model of networks. *WAW*, pages 44–55, 2004.
20. M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826, 2002.
21. C. Gkantsidis, M. Mihail, and E. W. Zegura. The markov chain simulation method for generating connected power law random graphs. *ALENEX*, pages 16–25, 2003.
22. S. L. Hakimi. On the realizability of a set of integers as degrees of the vertices of a graph. *SIAM J. Appl. Math*, 10, 1962.
23. V. Havel. A remark on the existence of finite graphs. *Caposis Pest. Mat.*, 80, 1955.
24. M. Jerrum and A. Sinclair. Fast uniform generation of regular graphs. *Theor. Comput. Sci.*, 73(1):91–100, 1990.
25. M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.
26. R. Kannan, P. Tetali, and S. Vempala. Simple markov-chain algorithms for generating bipartite graphs and tournaments. *Random Struct. Algorithms*, 14(4):293–308, 1999.
27. J. H. Kim and V. Vu. Generating random regular graphs. *Combinatorica*, 26(6):683–708, 2006.
28. J. Kleinberg. Small-world phenomena and the dynamics of information. *NIPS*, pages 431–438, 2001.
29. D. E. Knuth. The stanford graphbase: A platform for combinatorial computing, 1993.
30. T.G. Kolda, C. Seshadri, and A. Pinar. The bter graph model: Blocked two-level erds-rnyi. *Graph Exploitation Workshop, MIT*, 2011.
31. R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Random graph models for the web graph. *FOCS*, pages 57–65, 2000.
32. A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):1–5, October 2008.
33. J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research (JMLR)*, 11:985–1042, 2010.
34. J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. *KDD*, pages 177–187, 2005.
35. J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Statistical properties of community structure in large social and information networks. *WWW*, pages 695–704, 2008.
36. D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, 2003.
37. P. Mahadevan, C. Hubble, D. V. Krioukov, B. Huffaker, and A. Vahdat. Orbis: rescaling degree correlations to generate annotated internet topologies. *SIGCOMM*, pages 325–336, 2007.
38. P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat. Systematic topology analysis and generation using degree correlations. *SIGCOMM*, pages 135–146, 2006.
39. B. McKay. Asymptotics for symmetric 0-1 matrices with prescribed row sums. *Ars Combinatoria A*, 19:15–25, 1985.
40. M. Mihail and C. Papadimitriou. On the eigenvalue power law. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, RANDOM '02, pages 254–262, London, UK, 2002. Springer-Verlag.
41. M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, 2001.
42. M. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70(5):056131, Nov 2004.
43. M. Newman. Modularity and community structure in networks. *PNAS*, 103:8577–82, 2006.
44. M. E. J. Newman. The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. USA*, 98:404–409, 2001.
45. M. E. J. Newman. The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. USA*, 98:404–409, 2001.

46. M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Letter*, 89:208701, May 20 2002.
47. M. E. J. Newman. Mixing patterns in networks. *Phys. Rev. E.*, 67:026126, February 04 2002.
48. M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices, 036104. *Phys. Rev. E*, 74, 2006.
49. M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices, 036104. *Phys. Rev. E*, 74, 2006.
50. M. E. J. Newman. A symmetrized snapshot of the structure of the internet at the level of autonomous systems, reconstructed from bgp tables posted by the university of oregon route views project. *Unpublished*, July 22, 2006.
51. D. M. Pennock, G. Flake, S. Lawrence, E. J. Glover, and C.L. Andgiles. Winners dont take all: Characterizing the competition for links on the web. *Proceedings of the National Academy of Sciences*, 99:5207–5211, 2002.
52. A. E. Raftery and S. M. Lewis. The number of iterations, convergence diagnostics and generic metropolis algorithms. *Practical Markov Chain Monte Carlo*, pages 115–130, 1995.
53. E. Ravasz and A. L Barabasi. Hierarchical organization in complex networks. *Phys. Rev. E*, 67:026112, 2003.
54. A. Sala, S. Gaito, G. P. Rossi, H. Zheng, and B. Y. Zhao. Revisiting degree distribution models for social graph analysis. *http://arxiv.org/abs/1108.0027*, 2011.
55. A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Inf. Comput.*, 82(1):93–133, 1989.
56. A. Sokal. Monte carlo methods in statistical mechanics: Foundations and new algorithms. *Cous de Troisi‘eme Cycle de la Physique en Suisse Romande, Lausanne*, 1996.
57. A. Steger and N. C. Wormald. Generating random regular graphs quickly. *Combinatorics, Probability & Computing*, 8(4), 1999.
58. R. Taylor. Constrained switching in graphs. *SIAM J. ALG DISC. METH.*, 3, 1:115–121, 1982.
59. Z. Toroczkai and K.E. Bassler. Network dynamics: Jamming is limited in scale-free systems. *Nature*, 428:716, 2004.
60. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.
61. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.
62. J. G. White, E. Southgate, J. N. Thompson, and S. Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Phil. Trans. R. Soc. London*, 314:1–340, 1986.
63. W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

8 Appendix

Designing Synthetic Data

Our goal was to represent all of the potential means for $\frac{i}{20}$ for $0 < i \leq 20$. We note that 20 factors into 4 and 5, so we want to first fix some degrees such that $\mathscr{D}_k = 4$ and $\mathscr{D}_l = 5$. For convenience, because the maximum number of edges we will be assigning is 20, we will pick these degrees to be $K = \{20, 21, 22, 23, 24\}$ for $\mathscr{D}_k = 4$ and $L = \{25, 26, 27, 28\}$ for $\mathscr{D}_l = 5$. The number of each we picked was to guarantee that there were at least 20 combinations of edge types. We can now assign the values $1-20$ arbitrarily to $\mathscr{J}_{K\times L}$. This assignment clearly satisfies that $\mathscr{J}_{k,l} \leq \mathscr{D}_k\mathscr{D}_l$ so far.

Now, we must fill in the rest of $\mathscr{J}$ so that $\mathscr{D}$ is integer valued for degrees. One way is to note that we should have 4×20 degree 20 edges. We can sum the number of currently allocated edges with one endpoint of degree 20, call this x and set $\mathscr{J}_{1,20} = 80 - x$. There are many other ways of consistently completing $\mathscr{J}$, such as assigning as many edges as possible to the $K \times K$ and $L \times L$ entries, like $\mathscr{J}_{20,21}$. This results in a denser graph. For the synthetic graph used in this paper, we completed $\mathscr{J}$ by adding all edges as $(1,20),(1,21)$ etc edges. We chose this because it was simple to verify and it also made it easy to ignore the edges that were not of interest.